CHAPTER 1 Introduction of the PAT™ System API

1. Introduction of the PAT™ System API

The Open Text Search Engine API, is a query based API. Application programs, user interfaces and other front-ends invoke text queries by passing the appropriate commands to PAT and receiving the responses.

The query language for the search engine is documented in the Open Text Query Language Reference Guide. This document describes the mechanics of interfacing to the search engine.

There are several advantages to this setup. The first is that because the front-end and the PAT program are communicating using pipes, the two programs can be on separate machines (the standard Unix networking software supports pipes between processes on different machines). This facility allows the front-end to run on a relatively low-powered workstation while PAT runs on a bigger server machine.

The second advantage with the PAT system is that because the command and replies are all in ASCII text, no special tools are needed to debug the interface. A programmer can easily print out the commands that the front-end normally sends to PAT, to the screen. Similarly, the programmer can manually start PAT and type commands to PAT directly from the keyboard and have the results printed on the screen. This usually provides a very fast method to debug interface development.

The third advantage with this system is that it can be configured with any number of filter programs on each pipe. These programs are very useful for logging the queries that are sent to PAT and replies that PAT sends back (e.g. for administrative or debugging purposes) without having to disconnect the front-end and PAT. As well, filters can arbitrarily transform the data, providing a way to customize a general front-end without having to change any of the front-end code.

1.1 PAT's Verbose and Quiet modes

There are two ways of using PAT. The first way is by running PAT directly from the command line in a shell. This is PAT's 'verbose' mode, in which PAT puts out a prompt ('>>') when it is ready for a command, and the results are printed out in human readable form:

```
PAT™ Text Database System, Release 4.0
              Copyright 1987-1993 by
  Open Text Corporation and University of Waterloo
>> bush
  1: 232 matches
>> pr sample
  1850984, ...  </P><P>The Bush administration has insisted the country will ..
  3182788, ..P><P>A senior Bush administration trade official said that while..
  3104715, ..e Corp._  sued Bush and other former Silverado directors for $200..
  7241304, .. by President Bush do not preclude military action, Nikkhah said..
  7241075, ..abia and Mr.  Bush is scheduled to go later on in the month impl..
  6689199, ..nturn. </P><P>Bush received a generally bleak assessment of the ..
   772799, ..sident George Bush's reversal of his "no new taxes" pledge three..
  6390140, ..day.  </P><P> Bush said he looks favorably on including Canada i..
   721459, ..greement that Bush suggested could eventually encompass Canada. ..
  6515333, ..sident George Bush was unable to get the crucial backing of the ..
```

That mode of running PAT is fine for humans, but the format of the output is a bit too unstructured for programs to easily process. For this reason, PAT has a 'quiet' mode, within which it does not print out wordy responses, but instead prints them in a terse and structured format. The above example would look like the following if PAT were started up in quiet mode:

```
bush
<SSize>232</SSize>pr sample
<PSet><Start>1850984</Start><Start>3182788</Start><Start>3104715</
Start>
<Start>7241304</Start><Start>7241075</Start><Start>6689199</Start><-
Start
772799</Start><Start>6390140</Start><Start>721459</Start><-
Start>6515333<
/Start></PSet>
```

A brief explanation of the above output is in order. First, notice that there is no wordy message about "PAT Text Database System"at the start of the session. Also, notice that there aren't even any prompts. In quiet mode, PAT starts up quietly and immediately waits for the first command. In our example, the first command is the word 'bush' (which searches for the string 'bush' in the database). There are 232 matches to the string 'bush' in the database. In verbose mode (i.e., the first example), PAT prints that out as '1: 232 matches'. In quiet mode, the corresponding result is '`<SSize>232</SSize>`'. This tagged format can be parsed more easily by a program than the verbose mode format. Also note that there is no newline character after the '`</SSize>`' tag.

The next command is the 'pr sample' command. In verbose mode, this command produces one line per match point of the sample of the result from the 'bush' query. Each line starts with the offset of the match followed by a line of text surrounding the matchpoint. In quiet mode, only the start offsets are printed. The entire set of start offsets is surrounded by '`<PSet>`' (Pointer Set) and '`</PSet>`' tags. Then, each individual result is surrounded by '`<Start>`' and '`</Start>`' tags. This format is very regular and allows a program to easily identify both the content (i.e., the offsets) and the structure.

The following section further explains PAT's Quiet Mode.

1.2 Quiet Mode

{**QuietOn** Raw Converted Label Persistent}
{**QuietOff**}

changes the mode of operation of PAT. {**QuietOn**} causes PAT to operate in quiet mode. {**QuietOff**} causes PAT to revert to standard (non-quiet) mode.

Each of the four arguments to **QuietOn** is optional and may appear in any order. When an argument is present in a **QuietOn** command, the corresponding setting is turned on. Conversely, when an argument is not present in a **QuietOn** command, the corresponding setting is turned off. Settings are not carried forward from one **QuietOn** command to the next but are reset with each **QuietOn** command.

All PAT commands that create sets operate the same way in quiet mode and in standard mode. However, the output generated by PAT is different in the two modes. No prompt or newline appears when PAT is operating in quiet mode. In addition, the output from PAT in quiet mode is in a tagged format.

In standard mode, when a command or query creates a new set, a set number and the number of matches is output. In quiet mode, the tagged output contains the number of matches within `<SSize>` tags but no set number. For example, if a set of 122 matches is created by a PAT query the output is of the form:

```
<SSize>122</SSize>
```

In standard mode, information displayed about a set by a **pr** command is affected if a modifier is attached to the command. The output from a **pr** command is preceded by the offset in the text of the set member being printed. If the set is a region set, the offset is the start of each region in the set. In quiet mode, the output contains the numeric offset in a tagged format. The settings of Raw, Converted, Label and Persistent affect the information displayed by the **pr** command. Each of the settings is discussed below.

{QuietOn}

With Persistent turned off, the values of the offsets that are output are the logical offsets into the file. The logical and persistent offsets are different and non- interchangeable. Persistent offsets are designed for use with the update system. (See the documentation for the PAT update system.)

If the **pr** command is not of the form **pr.region**, the offset of each set member is contained within `<Start>` tags and the entire output is contained within `<PSet>` (for Point Set) tags. For example, if the set is of size 2, the output might look as follows (without the line breaks).

```
<PSet><Start>1234</Start>
<Start>5554</Start></PSet>
```

If the modifier to the **pr** command is **.region**, in standard mode the text displayed is from the match point to the end of a specified region. In quiet mode, the tagged output contains both the offset of the match point and the offset of the end of the specified region. The offsets of the ends of the region are contained within <End> tags and the entire output is contained within <RSet> (for Region Set) tags. For example, for the above set of size 2, output from a **pr.region** might look like

```
 <RSet><Start>1234</Start><End>1444</End>
<Start>5554</Start><End>6000</End></RSet>
```

{Quiet On Label}

When Label is turned on, the form in which the offset is printed changes. The numeric value of the offset into the text within the `<Start>` or <End> tags is preceded by an identifying label and a colon. This label string is the value of the **Label** setting. If **Label** has not been set, the label used in the output is the name of the data dictionary file up to the first non-alphanumeric character. For example, if the data dictionary is "news.dd" and **Label** has not been set, the output from a **pr** command would look like:

```
<PSet><Start>news:1234</Start>
<Start>news:5554</Start></PSet>
```

{QuietOn Raw }

When Raw is turned on, in addition to the tagged offsets, the output contains text showing the match point and surrounding context. For each member of the set, this additional information is output within <Raw> tags following the tagged offset information for each member of the set.

The length of the string being output is given within `<Size>` tags and is followed by the text. As in standard mode, if **pr** has no modifier, the length of string output is determined by the **PrintLength** setting and the context shown to the left of the match point is determined by the **LeftContext** setting. If the modifier is a numeric value, this value determines the length of the string and the left context is still determined by the **LeftContext** setting. If the modifier to the **pr** command is **.region**, the text starts at the match point and continues to the end of the specified region.

For example, assuming a **PrintLength** setting of 25, and a **LeftContext** setting of 5, the output from a **pr** command applied to a set of 2 matches to the string "sample" would be (without the line breaks shown here):

```
<PSet><Start>1234</Start><Raw><Size>25</Size>
This sample is to be firs</Raw>
<Start>3456</Start>
<Raw><Size>25</Size>
This sample is to be seco</Raw>
</PSet>
```

If the **SortOrder** setting is "OccurHead", in addition to the above output, the descriptive header is output in a tagged format. (See the entry for **SortOrder** for a description of the header). This information is contained within <Hdr> tags and includes the length of the descriptive string within `<Size>` tags followed by the string of the header. If the **SortOrder** setting was "OccurHead" in the above example, the output would be (without the line breaks shown here):

```
<PSet><Start>1234</Start>
<Hdr><Size>10</Size>First     </Hdr>
<Raw><Size>25</Size>
This sample is to be firs</Raw>
<Start>3456</Start>
<Hdr><Size>10</Size>Second    </Hdr>
<Raw><Size>25</Size>
This sample is to be seco</Raw>
</PSet>
```

{QuietOn Converted}

When Converted is turned on, in addition to the tagged offsets, text following the match point is output for each member of the set. This text is displayed with the appropriate character mappings for the PAT index and any stopwords removed. For example, if upper case is mapped to lower case when creating the index, the text is displayed in lower case. If the index has the word "to " as a stopword, "to " would not appear in the converted text.

For each member of the set, this additional information is output within `<Cvt>` tags. The length of the output text string is enclosed within `<Size>` tags and is followed by the text itself. For each set member, the text string shown starts at the match point. This is in contrast to the Raw text output which shows the match point with some left context. If the **pr** has no modifier, the length of the string is determined by the **PrintLength** setting. If the modifier is numeric, this determines the string length. If the modifier is **.region**, the length of the string is the value of the difference between the offsets of the match point and the end of the region. As the displayed text is converted text, it is possible that some text conversions cause output, such as multiple blanks resulting from character mappings or stopwords, to be suppressed. This may result in text that occurs past the end of the region to be displayed.

For example, using the above example of a set of size 2 and further assuming that "to" and "be" are stop-words the output might be:

```
<PSet><Start>1234</Start>

<Cvt><Size>25</Size>
sample is first used for </Cvt>
<Start>3456</Start>
<Cvt><Size>25</Size>
sample is second used for</Cvt>
</PSet>
```

If the **SortOrder** setting is "OccurHead", in addition to the above output, the descriptive header is given in a tagged format. (See the entry for **SortOrder** for a description of the header). The information giving the descriptive string precedes the `<Cvt>` tag and does not have the character mappings applied to it. The previous example would change to:

```
<PSet><Start>1234</Start>
<Hdr><Size>10</Size>First    ..<Cvt><Size>25</Size>
this sample is first used</Cvt>
<Start>3456</Start>
<Hdr><Size>10</Size>Second    ..<Cvt><Size>25</Size>
this sample is second use</Cvt>
</PSet>
```

{QuietOn Persistent}

When Persistent is turned on, the offsets that are output are the persistent (persistent) positions within the text database. As noted earlier, in a database that has not been initialized for update, the persistent and logical offsets are identical.

{QuietOn Raw Converted Label}

Any combination of the **QuietOn** arguments may be used. Thus, after the command **{QuietOn**Raw Converted Label**}**, the following would result:

```
<PSet><Start>news:1234</Start>
<Raw><Size>25</Size> This sample is to be firs</Raw>
<Cvt><Size>25</Size> sample is used first for </Cvt>
<Start>news:3456</Start>
<Raw><Size>25</Size> This sample size is to be seco
</Raw>
<Cvt><Size>25</Size>
sample is second used for</Cvt>
</PSet>
```